



09/520,684.

CJC

PATENT
Attorney Docket No. 27776

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

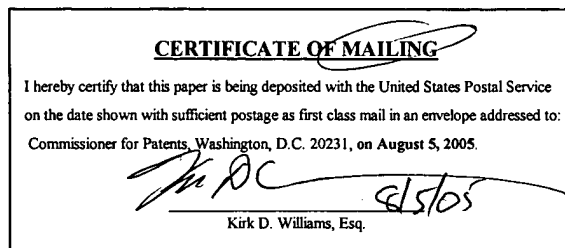
Patent No. 6,907,041

Confirmation No. 9090

Issued: Jun. 14, 2005

Name of Patentee: Turner et al.

Patent Title: COMMUNICATIONS
INTERCONNECTION NETWORK WITH
DISTRIBUTED RESEQUENCING



**REQUEST FOR CERTIFICATE OF CORRECTION OF
PATENT FOR PATENT OFFICE MISTAKE (37 C.F.R. § 1.322)**

Attn: Certificate of Correction Branch
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate
AUG 1 2 2005
of Correction

Dear Sir:

It is requested that a Certificate of Correction be issued to correct Office mistakes found the above-identified patent. Attached hereto is a Certificate of Correction which indicates the requested correction. For your convenience, also attached are copies of selected pages (a) from the issued patent with errors highlighted, (b) from the original application as filed March 7, 2000, (c) Amendment A filed June 21, 2003, and (d) Amendment D filed November 18, 2004 with the correct text/instructions.

AUG 1 5 2005


In re US Patent No. 6,907,041

It is believed that there is no charge for this request because applicant or applicants were not responsible for such error, as will be apparent upon a comparison of the issued patent with the application as filed or amended. However, the Assistant Commissioner is hereby authorized to charge any fee that may be required to Deposit Account No. 501430.

Respectfully submitted,
The Law Office of Kirk D. Williams

Date: August 5, 2005

By



8/5/05

Kirk D. Williams, Reg. No. 42,229
One of the Attorneys for Applicants
CUSTOMER NUMBER 26327
The Law Office of Kirk D. Williams
1234 S. OGDEN ST., Denver, CO 80210
303-282-0151 (telephone), 303-778-0748 (facsimile)

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,907,041

DATED : Jun. 14, 2005

INVENTOR(S) : Turner et al.

It is certified that error(s) appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Col. 1, line 5, replace "IVENTIONS" with -- INVENTIONS --

Col. 2, line 54, replace "list" with -- lists --

Col. 2, line 57, replace "an" with -- can --

Col. 2, line 58, replace "list" with -- lists --

Col. 2, line 60, replace "genely" with -- generally --

Col. 5, line 55, replace "purposes," with -- purposes; --

Col. 7, lines 26-27, replace "single-buffer" with -- single buffer --

Col. 8, line 37, replace " $10g_2 T$ " with -- $\log_2 T$ --

Col. 11, line 42, replace "here," with -- here; --

Col. 16, line 34, replace "funding" with -- finding --

Col. 16, line 34, replace "timestanp" with -- timestamp --

MAILING ADDRESS OF SENDER:

Kirk D. Williams, Reg. No. 42,229
Customer No. 26327
The Law Office of Kirk D. Williams
1234 S. Ogden Street, Denver, CO 80210

PATENT NO. 6,907,041

No. of additional copies

⇒ NONE (0)

AUG 15 2005

1

COMMUNICATIONS INTERCONNECTION NETWORK WITH DISTRIBUTED RESEQUENCING

2

INVENTIONS STATEMENT AS TO RIGHTS TO INVENTIONS
MADE UNDER UNIVERSITY SPONSORED
RESEARCH AND DEVELOPMENT

The invention was performed in the course of work under sponsorship of Washington University of St. Louis, Mo., and Growth Networks Inc., of Mountain View, Calif. Each entity claims joint ownership in the invention. Growth Networks Inc. has exclusive rights under license from Washington University.

FIELD OF THE INVENTION

This invention relates to communications information routing; and more specifically to switch interconnection networks or switch fabrics.

BACKGROUND OF THE INVENTION

Buffered multistage interconnection networks are often used in Asynchronous Transfer Mode ("ATM") and other types of packet switching systems. Networks of this type use buffers to store packets at intermediate points when contention for output links prevents immediate transmission. As used herein, the term packet is used to indicate generically addressable data packets of all types, including fixed length cells and variable length packets.

Many multistage interconnection networks provide multiple paths between network inputs and outputs, allowing the traffic to be balanced across the alternative paths. An example of such a network is shown in FIG. 1A, which is a depiction of an architecture known as a three stage Beneš network. The Beneš network 10 is composed of three stages of switch elements (SE) 12, 14, 16 and two webs of interconnecting links (IL) 18, 20. Switch elements 12, 14, 16 may have any number of input type interconnecting links 24 and output type interconnecting links 26. Letting d denote the number of input links and the number of output links in a single switch element and letting n denote the number of input links and output links of the multistage network as a whole, FIG. 1A illustrates $d=4$ and $n=16$. Traffic distribution in a multistage network 10 is commonly done in one of two ways for a given end-to-end session of sending packets from an identified input to an identified output of the network. In systems that use static routing, all packets associated with a given session follow the same path through the interconnection network. (In ATM networks, a session is typically associated with a virtual circuit.) This path is selected when the session begins and typically remains fixed until the session is completed, thus guaranteeing that packets belonging to the same session are forwarded in the order they arrived.

In systems that use dynamic routing, traffic is distributed on a packet-by-packet basis so as to equalize the traffic load across the entire interconnection network. Dynamic routing systems can distribute traffic more evenly than systems that use static routing. Consequently, dynamic routing systems can be operated with lower speed internal links than are needed for systems that use static routing. However, because dynamic routing systems do not constrain the packets belonging to a single user session to a single path, they may allow packets in a given session to get out of order, that is, lose sequence.

Systems using dynamic routing typically provide resequencing mechanisms to restore the correct packet order at

the outputs of the switching system. Conventional mechanisms for resequencing usually ensure that packets are delivered in the correct order, but under unusual conditions, they can fail to reorder packets correctly.

A known method for resequencing ATM packets in a multistage interconnection network uses timestamps and time-ordered output queues. Such systems require a central timing reference. The central timing reference signal is distributed to the circuits at all inputs to the network and to the circuits at all outputs. When a packet of data enters the network at an input, the current time is inserted into a timestamp field in the packet. When the packet emerges from the network at the appropriate destination, the timestamp is used to insert the packet into a time-ordered queue. In other words, the packets are read from the queue in increasing order of their timestamp values. Associated with the queue is an age threshold or timeout which specifies the minimum time that must elapse between the time a packet entered the interconnection network and the time it is allowed to leave the resequencing buffer at the output. If the difference between the current time and the timestamp of the first packet in the buffer is smaller than the age threshold, then the first packet is held back (along with all others "behind" it in the buffer). This allows packets that are delayed for a long period of time in the interconnection network to catch up with other packets that experience smaller delays.

If the age threshold is larger than the maximum delay that packets ever experience in the interconnection network, then the time-based resequencing method will always deliver packets in order. However, if packets are delayed by more than the time specified by the age threshold or timeout, errors may occur. In typical systems, delays in the interconnection network are usually fairly small (a few packet times per stage) and packets only rarely experience long delays. On the other hand, a worst-case delay may be very large. As a result, the age threshold is usually set not to the worst-case delay, but to a smaller value chosen to give an acceptably small frequency of resequencing errors. The resequencing delay can be traded off against reduced probability of resequencing errors. Conventional time-based resequencing methods can be implemented in various ways.

Merge sorting, illustrated in FIG. 1B, is a known technique for producing a single sorted list from multiple ordered lists whose values are known a priori. For example, two lists 920 and 922 of known elements sorted in ascending order can be combined into a single sorted list 924 by repeatedly taking the smaller value from the front of lists 920 and 922, and appending the smaller value to the end of list 924. This example can be extended to a set of n known values, which can be sorted by first dividing the set into n lists containing a single value each, then combining pairs of lists to produce $n/2$ lists with two values each. Pairs of these lists are then merged, producing $n/4$ lists with four values each. Continuing in this fashion eventually yields a single sorted list containing the original values, but in sorted order, as shown in FIG. 1B. Merge sorting can also be implemented using three-way merging (that is, merging three sorted lists into a single sorted list in one step), rather than by using two-way merging. More generally, d -way merging can be used for any integer $d > 1$.

However, known sorting techniques are not candidates for resequencing packets in a packet switching environment. In such a dynamic environment, lists of known values are not available. Rather, these packet switching systems must resequence streams of packets. Moreover, in dynamic resequencing systems, a complete set of packets to be resequenced

lists
can
lists
generally,

5

126, 127 can be combined into a single sorted stream of data packets in Stage 1 FIFO buffer 134 by repeatedly taking the packet with the smallest timestamp available from the buffers 126 and 127 and inserting it into the buffer 134. Both packets shown in Stage 1 FIFO buffer 134 in FIG. 2B came from source 127 in FIG. 2A, because both of these packets had smaller timestamps than any from source 126. This is not an uncommon occurrence.

Proceeding in this fashion, an arbitrary set of packets distributed across n source buffers can be sorted by selecting the packet with the smaller timestamp in each pair of source buffers 126, 127; 128, 129; 130, 131; 132, 133, as shown in FIG. 2A, and forwarding that packet to one of $n/2$ Stage 1 FIFO buffers 134, 135, 136, 137 as shown in FIG. 2B. The process is repeated in $n/4$ Stage 2 FIFO buffers 120, 122 as shown in FIG. 2C and completed in a FIFO buffer 124 as shown in FIG. 2D, where the first n packets have been stored in sorted order. Note that packets with later timestamps continue to flow into the system and are sorted in a similar fashion in their turn, as FIGS. 2A–D illustrate. FIGS. 2A–2D show a very simplified embodiment of the invention with only a single output and with no indication of the timing or control of the merging streams of timestamps. Other embodiments of the invention provide for multiple outputs and/or accommodating the timing or control of the merging streams of timestamps.

Certain embodiments of the invention further sense empty lists by means of a status message. In one such embodiment, a status message is an inserted packet having a timestamp value, as hereafter explained. A controller uses the timestamp value of the inserted packet to assure that all members of a set of source data packets destined for the same destination buffer have been processed. Embodiments of the invention also simultaneously sort and merge, within a switching element, a multiplicity of sorted streams of packets flowing on an input line into a multiplicity of sorted streams of packets flowing on an output line. The locally-performed sorting is further distributed among a plurality of switching elements forming a multistage interconnection network, so that packets received from a plurality of network sources are properly resequenced at each of a plurality of network destinations at the other side of the multistage interconnection network.

1. A Multistage Network with Distributed Resequencing

FIG. 3 illustrates an exemplary multistage network that implements distributed resequencing in accordance with the invention. A set of source buffers 50, 52, 54, 56 is followed by a three-stage network made up of four-port ($d=4$) switch elements 60, 62, 64, 66; 70, 72, 74, 76; and 80, 82, 84, 86. Each switch element has a two-slot arrival buffer 90 at each of its input ports and a four-slot departure buffer 92 at each of its output ports. Each buffer slot 96, 98, 100, 102, 104, 106 can contain a single fixed length packet, or cell. Small buffers and fixed length packets (known as cells) are used here for illustration purposes; some embodiments typically have more slots per buffer and may provide for variable length packets. In FIG. 3, the presence of a cell in a buffer slot, for instance, slot 98, is indicated by a pair of numbers. The top number in each pair is a value representing an internal address, namely the designation number of the network destination to which the cell is to be delivered. The bottom number in each pair in a slot is a value representing the cell's timestamp, which is the time when the cell passed from the source buffer 50, 52, 54, or 56 (where it originated) to the first stage switch element 60, 62, 64, or 66. If a buffer contains no cells, a status message packet substitute, which is simply a timestamp value without an output designation

6

number will be generated and hereinafter called a floor indication. Floor indications are shown in FIG. 3 as a pair of dashes, as in slots of elements 52, 60, 62, 66, 70, 72, 76, 80 (slot 110), 84, and 86, and a number, indicating the timestamp value. The presence of a floor indication with value j in a buffer means that any cells arriving in the future are guaranteed to have a timestamp at least as large as j .

FIG. 7 is a flowchart illustrating the steps performed in an embodiment of the invention by each switch element during a cell time, defined as the time required for one cell to be transferred from the output of a switch element to a downstream switch element. FIG. 4 shows the state of the network of FIG. 3 one cell time later. At the start of a cell time, each switch element examines its arrival buffers (Step A). For each arrival buffer that is not full, the switch element transmits a grant signal to the appropriate upstream neighbor element, indicating that it is safe for the upstream neighbor element to transmit a cell (Step B). No cells are transmitted between elements unless a grant signal is received. Other flow control techniques may be used.

Every departure buffer in the upstream neighbor element that has received a grant signal and has at least one cell sends its first cell. Every departure buffer in the upstream neighbor element that has received a grant signal but has no cell sends a floor indication. The timestamp value of the floor indication is the minimum of the timestamps of cells in the upstream neighbor element arrival buffers. For example, FIG. 3 shows that in switch element 70, the minimum timestamp in arrival buffers 200, 201, 202, 203 is "5"; thus, a floor indication with timestamp "5" has been placed in empty departure buffers 205, 206, 207. Buffer 208 is the only departure buffer containing a cell that can be propagated.

Source buffers 50, 52, 54, 56 send cells to first stage switching elements 60, 62, 64, 66 as long as there is room in the arrival buffers of these switching elements. If a source buffer is empty, a floor indication is sent; in a practical embodiment the timestamp value of the floor indication is simply the current time at that input port.

These cells and floor indications are received by the switch element (Step C). For instance, the cell in slot 106 of departure buffer 92 in FIG. 3 has moved to slot 220 of arrival buffer 200 in FIG. 4. When a floor indication arrives at a non-full arrival buffer, it is stored at the end of the arrival buffer, but if the arrival buffer already contains a floor indication as the last element, then the newly arrived floor indication replaces the old one. If a cell arrives at an arrival buffer containing a floor indication as the last element, the cell replaces the floor indication.

The switch element then performs a processing cycle (Steps D, E, F, G, H). The switch element determines the smallest timestamp value associated with any cell or floor indication in its arrival buffers (Step D). For example, in FIG. 3, switch element 82 determines that the smallest timestamp in its arrival buffers 160, 164, 166, 168 is "5."

The switch element next determines whether there is a cell having this smallest timestamp (Step E), for example, the cell in slot 162 of switch element 82. If there is a cell having the smallest timestamp, then the switch element moves the cell to an appropriate departure buffer (Step F); for example, the cell in slot 162 in FIG. 3 has moved to departure buffer 170 in FIG. 4. If this smallest timestamp is associated only with a floor indication, then processing is discontinued until the next cell time. For example, in switch element 80 in FIG. 3, the smallest timestamp in any arrival buffer is "2," which appears only in the floor indication in slot 110, and FIG. 4 shows that no cells have been moved

purposes;

from any arrival buffer in switch element 80. The floor indication "2" appears in empty departure buffers 112, 114, 115 in FIG. 4.

If multiple arrival buffers share the smallest timestamp value, as is the case for example in switch elements 72 and 86 in FIG. 3, then each of these arrival buffers is considered in turn. If the arrival buffer contains a cell, then that cell is advanced to the appropriate departure buffer. If instead the arrival buffer contains a floor indication, nothing is moved from that buffer. For example, in switch element 72, the packets in slots 190 and 192 in switch element 72 (FIG. 3) have moved to departure buffers 194 and 196, respectively, (FIG. 4); in switch element 86, the floor indication in slot 191 in FIG. 3 has not moved in FIG. 4, but it has been overwritten by a new incoming cell. Moreover, the cell in slot 193 in FIG. 3 has moved to output buffer 195 in FIG. 4. Because the buffers in the switch element have limited capacity, no cell will be forwarded from an arrival buffer if the required departure buffer has no space available.

The selection of the appropriate departure buffer depends on the stage in the network at which the switch element is located. In a three-stage network, a typical objective in the first stage is to distribute the traffic as evenly as possible. Thus, a switch element in the first stage 60, 62, 64, 66 routes cells to a departure buffer containing the smallest number of cells, selecting a departure buffer randomly when no single-buffer contains fewer cells than all others. In the second and third stages, a switch element 70, 72, 74, 76; 80, 82, 84, 86 routes each cell to the departure buffer for the outgoing link that lies on the path to the destination output port specified in the cell header.

Referring to FIG. 7, if moving a cell to a departure buffer leaves any of the switch element's arrival buffers empty during the processing cycle, the switch element places a floor indication in the empty arrival buffer (Step G). The timestamp of this floor indication is equal to the timestamp of the last cell that was taken from the buffer.

If time remains in the cell time (Step H), the switch element may perform additional processing cycles. In a practical embodiment a d-port switch element performs at least d processing cycles (Steps D, E, F, G, H) during a single cell time, unless processing is stopped due to a floor indication having the minimum timestamp.

Embodiments of the invention include networks that handle variable sized packets and networks with more than three stages. For example, in a five-stage Beneš or Clos network, typically the first two stages perform the traffic distribution function, while the last three route packets based on their destination. In general, the first k-1 stages of a 2k-1 stage network perform traffic distribution, while the last k stages route packets based on their destination. This method can of course also be applied to networks in which the switch elements have different dimensions (numbers of inputs and outputs) and different buffer capacities and organizations.

The description above focuses on switch elements in which the arrival buffers and departure buffers are all of fixed size, and there is no sharing of memory space. However, in keeping with the scope and spirit of the invention, distributed resequencing can also be applied to switch elements in which some of the storage space is shared, rather than reserved for the exclusive use of particular inputs or outputs. It is required that each buffer have at least one dedicated storage slot, in order to prevent deadlock. The required modifications of the above procedures are straightforward.

To support multicast communication, wherein copies of a given packet are sent to multiple outputs (multipoint communication), the procedure for transferring packets from arrival buffers to departure buffers must be modified. In particular, a packet of minimum timestamp must be forwarded to all departure buffers on paths to its destinations. This need not involve storing multiple copies of the packet. It is sufficient to place a pointer to the stored packet in each of the appropriate departure buffers and store a reference count with the packet, so that the storage space can be released after the last required copy is sent. A switch element can use a variety of methods to decide to which local departure buffers to forward a multicast packet to, including the use of explicit information stored in the packet header, and/or the use of a multicast routing table with the switch element.

So long as the different inputs assign timestamps to packets in increasing order, the collective operation of the system guarantees that the packets delivered at any output of the last stage switch elements will have non-decreasing timestamps.

The timestamps assigned by different network inputs do not have to be strictly synchronized. If one input is slightly ahead of another, this simply means that its packets will be held back a little extra time by the network, so that they can be delivered in timestamp order.

Because the timestamps must be represented using a finite number of bits, certain embodiments of the distributed resequencing methods and apparatus must allow for the possibility of timestamp values that wrap around due to the constraints of a finite representation. This problem can be avoided if the network is operated so that no packet can be delayed more than some maximum time T from the time it first arrives at a source buffer to the time it is delivered to its destination. So long as there is such a maximum delay, then if more than $\log_2 T$ bits are used to represent the timestamps, a time value can always be unambiguously interpreted by comparing it to the current time.

2. Systems with Multiple Priorities

In an embodiment of the invention, preferential treatment can be provided for some traffic relative to other traffic by assigning each packet passing through the system a priority and by handling packets with different priority values differently. Systems using distributed resequencing can accommodate multiple priority classes in accordance with the scope and spirit of the invention. In an embodiment system with m different priority classes, each buffer in FIG. 3 is replaced by a collection of m buffers, where each buffer contains packets of a different priority class. FIG. 8 shows an example four-port switch element 800 that supports three priority classes. Each source port 801-804 can supply packets to any one of three buffers; for instance, source port 801 can supply packets to arrival buffers 811, 812, 813. Likewise, each output port 805-808 can forward packets from one of three buffers; for instance, output port 805 can forward packets from departure buffers 851, 852, and 853. In FIG. 8, a packet is represented by a pair of numbers separated by a period (e.g., in buffer 812) indicating the destination and the timestamp in the format <destination>.<timestamp> in each field where there is a destination. Floor indications are shown here as underlined numbers. Each buffer is labeled P1, P2, or P3 to indicate priority, with P1 being highest priority.

Many different embodiments of systems handling multiple priority classes are possible in keeping with the scope and spirit of the invention. One such exemplary embodiment

$\log_2 T$

11

buffers as possible during a given cell time. To provide fair treatment of all network outputs, the switch element can use round robin scheduling when selecting an active output during each processing cycle.

From time to time, floor indications must be sent from each switch element to its downstream neighbor elements for all empty buffers. In a large system with per output buffers, the number of floor indications that must be sent can become prohibitive. However, as will be discussed below, in the context of systems with reduced per output buffers, this problem can be avoided by combining and transmitting floor indications in a bandwidth-efficient manner.

The number of separate buffers required in a switch element can become large. For instance, if the network of FIG. 3 were modified to provide per output buffers, each first stage switch element 60, 62, 64, 66 would require 128 distinct buffers (64 arrival buffers and 64 departure buffers), while each second stage switch element 70, 72, 74, 76 would require 80 buffers (64 arrival buffers and 16 output buffers) and each third stage switch element 80, 82, 84, 86 would require 20 buffers (16 arrival buffers and 4 output buffers). In general, a multistage interconnection network with d -port switch elements, $2k-1$ stages, and $n=d^k$ requires $2dn$ distinct buffers in each of the first $k-1$ stages, $dn+n$ in the center stage, and $n+n/d$ in the stage following the center stage. Switch elements in subsequent stages require $1/d$ times the number of buffers as switch elements in the immediately preceding stage; thus, the last stage switch elements require d^2+d buffers each. To reduce memory requirements, switch element buffers can be implemented using linked list techniques so that only the list headers need to be replicated for each network output. This makes it practical for a switch element to implement several thousand distinct buffers. To ensure that deadlock cannot occur, each buffer must have at least one packet storage slot dedicated to it.

A system with per output buffers can also support m priority classes by further increasing the number of buffers. In particular, to convert a single priority system with per output buffers to a system supporting m priority classes requires that each buffer be replaced with m buffers, one for each class. The single priority case is discussed here, the extension to multiple priorities is straightforward.

4. Systems with Reduced Per Output Buffers

The number of distinct buffers needed for per output queuing can become large enough to limit the size of systems that can be built. Thus, it is significant that the number of buffers needed in each switch element can be substantially reduced by restricting the use of per output buffers to the final stages of the network, where they are most beneficial. An embodiment with reduced per output buffers is illustrated in FIG. 6, which shows a three-stage network with two-port switch elements. In this network, the first stage switch elements 360, 362 have just a single buffer per input port 363, 364 and a single buffer per output port 365, 366, as in the system of FIG. 3. In these switch elements, each packet is represented by a decimal number, as in buffer 363: the destination appears before the decimal point, and the timestamp appears after it. Floor indications are represented as underlined whole numbers, as in buffer 367. The second stage switch elements 370, 372 have a single arrival buffer per input port 373, 374, but have separate departure buffers for each network output 375-378. The third stage switch elements have separate arrival buffers 383-386 and departure buffers 387, 388 for each reachable output. In buffers dedicated to a single network output, for instance, buffer 378, the destination of each packet has been

12

suppressed. Note that the source buffers 350, 352, 354, 356 are still organized on a per output basis. In this system, the first stage switch elements operate like the first stage switch elements in the system of FIG. 3, while the third stage switch elements operate like the third stage switch elements in the system of FIG. 5. The center stage switch elements provide a transition from the shared buffers used in the first stage of the network to the per output buffers used in the third stage.

Providing effective traffic isolation in the system of FIG. 6 requires the introduction of a per output flow control mechanism to prevent departure buffers in the center stage switch elements from becoming full. In particular, when the buffers in the center stage switch elements for a particular network output become nearly full, a flow control signal is sent to all of the source buffers 350, 352, 354, 356, telling them to suspend transmitting packets to the congested output. For instance, if buffer 376 is close to being full, a signal is sent to all four network inputs telling them to suspend transmission of packets to output port 1. More precisely, there is a threshold defined for the departure buffers in the second stage switch elements. When the number of packets in a buffer exceeds the threshold, a flow control signal is sent, shutting off the flow of packets to the congested output. After the number of packets in the buffer drops below the threshold again, the network inputs are allowed to resume transmission of packets to the congested output. As long as flow control is exerted early enough to prevent buffers in the second stage switch elements from filling up, the system can maintain effective traffic isolation among the different outputs. This requires enough packet storage slots in each switch element to ensure that the buffer does not run out of space between the time the buffer level rises above the threshold and the time the inflow of packets to a congested output stops.

To implement this reduced per output buffering in the system of FIG. 3, each first stage switch element requires 8 separate buffers, each second stage switch element requires 20 buffers and each third stage switch element requires 20 buffers. To avoid deadlock, each of these buffers must have at least one dedicated packet storage slot. The approach can be generalized to systems with more than three stages. In a system with d -port switch elements, $2k-1$ stages, and $n=d^k$, the switch elements in the first $k-1$ stages of this network can use shared buffers, so each of these switch elements has $2d$ buffers. The switch elements from the middle stage onward can use per output buffers. In this case, each center stage switch element will have $d+n$ buffers, each switch element in the next stage will have $n+n/d$ buffers and switch elements in each subsequent stage will have $1/d$ times the number in the immediately preceding stage. The approach can also be generalized further to have per output buffering start later than the center stage. In this case, the per output flow control must be implemented at the stage where the transition from shared to per output buffers occurs.

With per output buffering, a switch element must send floor indications to its downstream neighbors for each of its departure buffers. The amount of information that must be exchanged because of this can become impracticably large. With reduced per output buffering, it is possible to greatly reduce the amount of floor information that must be sent by combining floor indications for different departure buffers in the form of floor vectors, which are generated by each switch element in the stage where the transition from shared buffers to per output buffers occurs, typically the center stage. In any one of these switch elements, for instance, in switch element 370 of FIG. 6, all empty departure buffers must have the same timestamp value for their floor indica-

here;

15

tamp value, always discontinuing forwarding of said one or more data packets for the remaining duration of a current cell time; and

in response to identifying that a particular data packet of said data packets has associated therewith the earliest timestamp value, forwarding the particular data packet during the current cell time; wherein said forwarding the particular data packet during the current cell time includes removing the particular data packet from an arrival buffer and if said removing causes the arrival buffer to become empty, in response adding a new floor indication to the arrival buffer.

2. The method of claim 1, wherein time remains in the current cell time to forward at least one of said one or more data packets when said discontinuing forwarding of said one or more data packets during the current cell time is performed.

3. An apparatus for distributed resequencing of packets in a packet switching system, the method comprising:

means for identifying one or more floor indications received by a switching element, each of said one or more floor indications associated with a respective timestamp value;

means for identifying one or more data packets received by the switching element, each of said one or more data packets associated with a respective timestamp value;

means for finding an earliest timestamp value associated with said one or more floor indications and said one or more data packets;

means for always discontinuing forwarding of said one or more data packets for the remaining duration a current cell time in response to making a determination that not one of said data packets has associated therewith the earliest timestamp value; and

means for forwarding the particular data packet during the current cell time in response to identifying that a particular data packet of said data packets has associated therewith the earliest timestamp value; wherein said means for forwarding the particular data packet during the current cell time includes means for removing the particular data packet from an arrival buffer and if said removing causes the arrival buffer to become empty, in response adding a new floor indication to the arrival buffer.

4. The apparatus of claim 3, wherein time remains in the current cell time to forward at least one of said one or more data packets when said discontinuing forwarding of said one or more data packets during the current cell time is performed.

5. A method for distributed resequencing of packets in a packet switching system, the method comprising:

16

identifying one or more floor indications received by a switching element, each of said one or more floor indications associated with a respective timestamp value;

identifying one or more data packets received by the switching element, each of said one or more data packets associated with a respective timestamp value; finding an earliest timestamp value associated with said one or more floor indications and said one or more data packets;

in response to making a determination that not one of said data packets has associated therewith the earliest timestamp value, always discontinuing forwarding of said one or more data packets for the remaining duration of a current cell time; and

in response to identifying that a particular data packet of said data packets has associated therewith the earliest timestamp value, forwarding the particular data packet during the current cell time.

6. The method of claim 5, wherein time remains in the current cell time to forward at least one of said one or more data packets when said discontinuing forwarding of said one or more data packets during the current cell time is performed.

7. An apparatus for distributed resequencing of packets in a packet switching system, the method comprising:

means for identifying one or more floor indications received by a switching element, each of said one or more floor indications associated with a respective timestamp value;

means for identifying one or more data packets received by the switching element, each of said one or more data packets associated with a respective timestamp value; means for finding an earliest timestamp value associated with said one or more floor indications and said one or more data packets;

means for always discontinuing forwarding of said one or more data packets for the remaining duration of a current cell time in response to making a determination that not one of said data packets has associated therewith the earliest timestamp value; and

means for forwarding the particular data packet during the current cell time in response to identifying that a particular data packet of said data packets has associated therewith the earliest timestamp value.

8. The apparatus of claim 7, wherein time remains in the current cell time to forward at least one of said one or more data packets when said discontinuing forwarding of said one or more data packets during the current cell time is performed.

* * * * *

finding
timestamp

COMMUNICATIONS INTERCONNECTION NETWORK WITH DISTRIBUTED RESEQUENCING

STATEMENT AS TO RIGHTS TO INVENTIONS MADE UNDER UNIVERSITY SPONSORED RESEARCH AND DEVELOPMENT

5

The invention was performed in the course of work under sponsorship of Washington University of St. Louis, Missouri, and Growth Networks Inc., of Mountain View, California. Each entity claims joint ownership in the invention. Growth Networks Inc. has exclusive rights under license from Washington University.

10

FIELD OF THE INVENTION

This invention relates to communications information routing; and more specifically to switch interconnection networks or switch fabrics.

15

BACKGROUND OF THE INVENTION

Buffered multistage interconnection networks are often used in Asynchronous Transfer Mode ("ATM") and other types of packet switching systems. Networks of this type use buffers to store packets at intermediate points when contention for output links prevents immediate transmission. As used herein, the term packet is used to indicate generically addressable data packets of all types, including fixed length cells and variable length packets.

20

Many multistage interconnection networks provide multiple paths between network inputs and outputs, allowing the traffic to be balanced across the alternative paths. An example of such a network is shown in Figure 1A, which is a depiction of an architecture known as a three stage Beneš network. The Beneš network 10 is composed of three stages of switch elements (SE) 12, 14, 16 and two webs of interconnecting links (IL) 18, 20. Switch elements 12, 14, 16 may have any number of input type interconnecting links 24 and output type interconnecting links 26. Letting d denote the number of input links and the number of output links in a single switch element and letting n denote the number of input links and output links of the multistage network as a whole, Figure 1A illustrates $d=4$ and $n=16$. Traffic

25

30

From Amendment A Filed 6-21-03

In re TURNER ET AL.
Application No. 09/520,684

IN THE SPECIFICATION:

Please replace the following paragraph beginning on page 4, line 18, with the following paragraph which is marked-up to indicate the changes:

Merge sorting, illustrated in Fig. 1B, is a known technique for producing a single sorted list from multiple ordered lists whose values are known a priori. For example, two lists 920 and 922 of known elements sorted in ascending order can be combined into a single sorted list 924 by repeatedly taking the smaller value from the ~~top~~ front of lists 920 and 922, and appending the smaller value to the end of list 924. This example can be extended to a set of n known values, which can be sorted by first dividing the set into n lists containing a single value each, then combining pairs of lists to produce $n/2$ lists with two values each. Pairs of these lists are then merged, producing $n/4$ lists with four values each. Continuing in this fashion eventually yields a single sorted list containing the original values, but in sorted order, as shown in Figure 1B. Merge sorting can also be implemented using three-way merging (that is, merging three sorted lists into a single sorted list in one step), rather than by using two-way merging. More generally, d -way merging can be used for any integer $d > 1$.

AUG 15 2005

the invention with only a single output and with no indication of the timing or control of the merging streams of timestamps. Other embodiments of the invention provide for multiple outputs and/or accommodating the timing or control of the merging streams of timestamps.

5 Certain embodiments of the invention further sense empty lists by means of a status message. In one such embodiment, a status message is an inserted packet having a timestamp value, as hereafter explained. A controller uses the timestamp value of the inserted packet to assure that all members of a set of source data packets destined for the same destination buffer have been processed.

10 Embodiments of the invention also simultaneously sort and merge, within a switching element, a multiplicity of sorted streams of packets flowing on an input line into a multiplicity of sorted streams of packets flowing on an output line. The locally-performed sorting is further distributed among a plurality of switching elements forming a multistage interconnection network, so that packets received from a
15 plurality of network sources are properly resequenced at each of a plurality of network destinations at the other side of the multistage interconnection network.

1. A Multistage Network with Distributed Resequencing

Figure 3 illustrates an exemplary multistage network that implements distributed resequencing in accordance with the invention. A set of source buffers 50, 52, 54, 56 is followed by a three-stage network made up of four-port ($d = 4$) switch
20 elements 60, 62, 64, 66; 70, 72, 74, 76; and 80, 82, 84, 86. Each switch element has a two-slot arrival buffer 90 at each of its input ports and a four-slot departure buffer 92 at each of its output ports. Each buffer slot 96, 98, 100, 102, 104, 106 can contain a single fixed length packet, or cell. Small buffers and fixed length packets (known as
25 cells) are used here for illustration (purposes); some embodiments typically have more slots per buffer and may provide for variable length packets. In Fig. 3, the presence of a cell in a buffer slot, for instance, slot 98, is indicated by a pair of numbers. The top number in each pair is a value representing an internal address, namely the designation number of the network destination to which the cell is to be delivered.
30 The bottom number in each pair in a slot is a value representing the cell's timestamp, which is the time when the cell passed from the source buffer 50, 52, 54, or 56 (where

indication in slot 191 in Fig. 3 has not moved in Fig. 4, but it has been overwritten by a new incoming cell. Moreover, the cell in slot 193 in Fig. 3 has moved to output buffer 195 in Fig. 4. Because the buffers in the switch element have limited capacity, no cell will be forwarded from an arrival buffer if the required departure buffer has no
5 space available.

The selection of the appropriate departure buffer depends on the stage in the network at which the switch element is located. In a three-stage network, a typical objective in the first stage is to distribute the traffic as evenly as possible. Thus, a switch element in the first stage 60, 62, 64, 66 routes cells to a departure
10 buffer containing the smallest number of cells, selecting a departure buffer randomly when no single buffer contains fewer cells than all others. In the second and third stages, a switch element 70, 72, 74, 76; 80, 82, 84, 86 routes each cell to the departure buffer for the outgoing link that lies on the path to the destination output port specified in the cell header.

15 Referring to Fig. 7, if moving a cell to a departure buffer leaves any of the switch element's arrival buffers empty during the processing cycle, the switch element places a floor indication in the empty arrival buffer (Step G). The timestamp of this floor indication is equal to the timestamp of the last cell that was taken from the buffer.

20 If time remains in the cell time (Step H), the switch element may perform additional processing cycles. In a practical embodiment a d -port switch element performs at least d processing cycles (Steps D, E, F, G, H) during a single cell time, unless processing is stopped due to a floor indication having the minimum timestamp.

25 Embodiments of the invention include networks that handle variable sized packets and networks with more than three stages. For example, in a five-stage Beneš or Clos network, typically the first two stages perform the traffic distribution function, while the last three route packets based on their destination. In general, the first $k-1$ stages of a $2k-1$ stage network perform traffic distribution, while the last k
30 stages route packets based on their destination. This method can of course also be

allow for the possibility of timestamp values that wrap around due to the constraints of a finite representation. This problem can be avoided if the network is operated so that no packet can be delayed more than some maximum time T from the time it first arrives at a source buffer to the time it is delivered to its destination. So long as there is such a maximum delay, then if more than $\log_2 T$ bits are used to represent the timestamps, a time value can always be unambiguously interpreted by comparing it to the current time.

2. Systems with Multiple Priorities

In an embodiment of the invention, preferential treatment can be provided for some traffic relative to other traffic by assigning each packet passing through the system a priority and by handling packets with different priority values differently. Systems using distributed resequencing can accommodate multiple priority classes in accordance with the scope and spirit of the invention. In an embodiment system with m different priority classes, each buffer in Figure 3 is replaced by a collection of m buffers, where each buffer contains packets of a different priority class. Figure 8 shows an example four-port switch element 800 that supports three priority classes. Each source port 801–804 can supply packets to any one of three buffers; for instance, source port 801 can supply packets to arrival buffers 811, 812, 813. Likewise, each output port 805–808 can forward packets from one of three buffers; for instance, output port 805 can forward packets from departure buffers 851, 852, and 853. In Figure 8, a packet is represented by a pair of numbers separated by a period (e.g., in buffer 812) indicating the destination and the timestamp in the format <destination>.<timestamp> in each field where there is a destination. Floor indications are shown here as underlined numbers. Each buffer is labeled P1, P2, or P3 to indicate priority, with P1 being highest priority.

Many different embodiments of systems handling multiple priority classes are possible in keeping with the scope and spirit of the invention. One such exemplary embodiment system with multiple priority classes differs in the following respects from a typical embodiment system with a single priority class.

If an output port of a given switch element has packets waiting in several of its buffers, one of the buffers is selected and the first packet in the selected

From time to time, floor indications must be sent from each switch element to its downstream neighbor elements for all empty buffers. In a large system with per output buffers, the number of floor indications that must be sent can become prohibitive. However, as will be discussed below, in the context of systems with
5 reduced per output buffers, this problem can be avoided by combining and transmitting floor indications in a bandwidth-efficient manner.

The number of separate buffers required in a switch element can become large. For instance, if the network of Figure 3 were modified to provide per output buffers, each first stage switch element 60, 62, 64, 66 would require 128
10 distinct buffers (64 arrival buffers and 64 departure buffers), while each second stage switch element 70, 72, 74, 76 would require 80 buffers (64 arrival buffers and 16 output buffers) and each third stage switch element 80, 82, 84, 86 would require 20 buffers (16 arrival buffers and 4 output buffers). In general, a multistage interconnection network with d -port switch elements, $2k-1$ stages, and $n=d^k$ requires
15 $2dn$ distinct buffers in each of the first $k-1$ stages, $dn+n$ in the center stage, and $n+n/d$ in the stage following the center stage. Switch elements in subsequent stages require $1/d$ times the number of buffers as switch elements in the immediately preceding stage; thus, the last stage switch elements require d^2+d buffers each. To reduce memory requirements, switch element buffers can be implemented using linked list
20 techniques so that only the list headers need to be replicated for each network output. This makes it practical for a switch element to implement several thousand distinct buffers. To ensure that deadlock cannot occur, each buffer must have at least one packet storage slot dedicated to it.

A system with per output buffers can also support m priority classes by
25 further increasing the number of buffers. In particular, to convert a single priority system with per output buffers to a system supporting m priority classes requires that each buffer be replaced with m buffers, one for each class. The single priority case is discussed here; the extension to multiple priorities is straightforward.

4. Systems with Reduced Per Output Buffers

30 The number of distinct buffers needed for per output queuing can become large enough to limit the size of systems that can be built. Thus, it is

From Amendment D (and RCE) Filed 11-18-2004

In re TURNER ET AL., Application No. 09/520,684
Amendment *CD and RCE*

36 (currently amended): An apparatus for distributed resequencing of packets in a packet switching system, the method comprising:

means for identifying one or more floor indications received by a switching element, each of said one or more floor indications associated with a respective timestamp value;

means for identifying one or more data packets received by the switching element, each of said one or more data packets associated with a respective timestamp value;

means for finding an earliest timestamp value associated with said one or more floor indications and said one or more data packets;

means for always discontinuing forwarding of said one or more data packets during for the remaining duration of a current cell time in response to identifying making a determination that not one of said data packets has associated therewith the earliest timestamp value; and

means for forwarding the particular data packet during the current cell time in response to identifying that a particular data packet of said data packets has associated therewith the earliest timestamp value.

37 (previously presented): The apparatus of claim 36, wherein time remains in the current cell time to forward at least one of said one or more data packets when said discontinuing forwarding of said one or more data packets during the current cell time is performed.